Multiple Sink and Relay Placement in Wireless Sensor Networks

Lanny Sitanayah [1] and **Kenneth N. Brown** [2] and **Cormac J. Sreenan** [3]

Abstract. Wireless sensor networks are subject to failures. Deployment planning should ensure that when a sink or sensor node fails, the remaining network can still be connected, and so may require placing multiple sinks and relay nodes in addition to sensors. For network performance requirements, there may also be path-length constraints for each sensor node. We propose two local search algorithms, GRASP-MSP and GRASP-MSRP, to solve the problem of multiple sink placement and the problem of multiple sink and relay placement, respectively. GRASP-MSP minimises the deployment cost, while ensuring that each sensor node in the network is double-covered, i.e. it has two length-constrained paths to two sinks. GRASP-MSRP deploys sinks and relays to minimise the deployment cost and to guarantee that all sensor nodes in the network are double-covered and noncritical. A sensor node is noncritical if upon its removal, all remaining sensor nodes still have length-constrained paths to sinks. We evaluate the algorithms empirically and show that both GRASP-MSP and GRASP-MSRP outperform the closely-related algorithms from the literature for the lowest total deployment cost.

1 INTRODUCTION

A wireless sensor network (WSN) is composed of a large number of sensor nodes [1]. Each sensor node is a battery-powered device with limited storage, processing and communication capability. It is able to sense a close-by physical phenomenon, perform a simple computation and send its data wirelessly over a multi-hop network to a special node called a data sink. This network is subject to failure as the wireless devices and the communication links are unreliable. A sensor node may fail due to limited battery life or hardware malfunction, or may be damaged by weather or human intervention. When some sensor nodes fail, the network may be disconnected and thus it cannot gather information from the isolated area. Although a sink has more resources than a sensor node, this electronic device may fail too.

To protect against network failure, it is important to plan the topology deployment. In this paper, we study WSN deployment planning, with the aim of protecting the network against a single failure. That is, after a failure of a sink or a sensor node, each remaining sensor node can deliver its data to a sink through a multi-hop path with an acceptable length. We consider the path length restriction as data latency requirements may be important in WSN applications. To be robust to sink failure, it is necessary to deploy multiple sinks in the network such that each sensor node is *double-covered*, i.e. it has length-bound paths to two sinks. While we restrict our assumption to k-covered, where $k = 2$ in this paper, our solution is also applicable to any integer $k \geq 1$. To protect against sensor node failure, it is necessary to place some relay nodes, which do not sense, but only forward data from other nodes. So when a sensor node fails, each remaining sensor node still has at least one length-bound path to a sink. Installing sinks and relays comes at a cost that includes not only the hardware purchases, but also the installation and maintenance cost, thus motivating our solution to minimise the total deployment cost.

Our main contribution is a solution that minimises the total cost of sink and relay deployment but ensures the network robustness against a single device failure, either a sink or a sensor node. Our solution uses local search algorithms based on GRASP [4]. Firstly, we look at the multiple sink placement (MSP) problem to ensure that each sensor node is double-covered. We propose GRASP-MSP and show that it can find the same cost as the optimal solution with shorter runtime. Even though finding the optimal solution is sufficient for the multiple sink placement problem, the GRASP-MSP performance gives us confidence to use the same local search technique for the more complex multiple sink and relay placement (MSRP) problem. GRASP-MSRP employs the concept of length-constrained connectivity and rerouting centrality (l-CRC) introduced in [10] to identify every sensor node which if failed can cause other nodes to lose their length-bound paths to sinks. We demonstrate empirically that the solutions produced by GRASP-MSRP are over 30% less costly than those of a greedy-based approach. Although GRASP-MSRP's execution time is longer than that for the greedy approach, the runtime issue is not a problem since the deployment planning is an offline process that is executed before the actual deployment.

The remainder of this paper is organised as follows. In Section 2, we briefly describe the background and review the related work on sink and relay deployment algorithms. We present and evaluate GRASP-MSP and GRASP-MSRP in Section 3 and 4, respectively. Finally, Section 5 concludes the paper.

2 BACKGROUND AND RELATED WORK

A WSN can be modeled as a graph $G = (V, E)$, where V is a set of nodes and E is a set of edges. Each edge connects two nodes that are within transmission range of each other[4], and the two nodes are said to be *adjacent*. A *path* of length t between two nodes v and w is a sequence of nodes $v = v_0, v_1, \ldots, v_t = w$, such that v_i and v_{i+1} are adjacent for each i. A path from a node v to a set of nodes W is simply

[1] Mobile & Internet Systems Laboratory (MISL) and Cork Constraint Computation Centre (4C), Department of Computer Science, University College Cork, Ireland, email: ls3@cs.ucc.ie

[2] Cork Constraint Computation Centre (4C), Department of Computer Science, University College Cork, Ireland, email: k.brown@cs.ucc.ie

[3] Mobile & Internet Systems Laboratory (MISL), Department of Computer Science, University College Cork, Ireland, email: cjs@cs.ucc.ie

[4] For simplicity we assume bi-directional links, but this could be easily relaxed by specifying a more complex connectivity graph.

a path from v to any node $w \in W$. Two nodes are *connected* if there is a path between them. A graph is connected if every pair of nodes is connected. Sensor network topology is an undirected graph and for simplicity, we assume that the graph is connected. $H = (W, E{\downarrow_W})$ is an induced subgraph of $G = (V, E)$ if $W \subset V$ and $E{\downarrow_W}$ has exactly the edges that appear in G over the same vertex set (where $E{\downarrow_X}$ means a set of edges restricted to those that connect nodes in X).

In the literature, the problems of deploying sinks and relays are solved separately. Some sink deployment algorithms have been proposed with objectives to minimise and balance the energy consumption across networks [11, 6], reduce packet delivery latency [12], meet the required lifetime [8] and make the network double-covered for fault-tolerance [3]. Even though the algorithm in [3] is not designed for WSNs, the problem of finding the optimal positions for core nodes in passive optical networks [3] is similar to the problem of finding optimal positions for sinks in WSNs. Similar to our objective, i.e. to minimise the deployment cost, the algorithms proposed in [11, 8] try to minimise the number of sinks deployed, while the number of sinks is given in [6, 12, 3]. In [11], a sink is chosen greedily from a set of candidate locations such that it can cover as many sensor nodes, which are within the hop count bound from the sink, as possible. However, this algorithm does not consider double-covered networks. In [8], the algorithm deploys sinks one by one until the desired network lifetime is reached. It does not make the network double-covered and uses the well-known k-means clustering algorithm to identify the positions of sinks, which are assumed can be placed anywhere in the region. The algorithm in [3] also uses the k-means clustering algorithm to place a given number of sinks to make the network double-covered.

The problem of deploying relay nodes for increased reliability has long been acknowledged as a significant problem [2, 9, 5, 7, 10]. The relay placement algorithm proposed in [10], GRASP-ABP, is also a GRASP-based local search algorithm. It uses length-constrained connectivity and rerouting centrality (l-CRC) to identify critical nodes. Centrality indices is a core concept in social network analysis, used to determine the importance of a node in a network. Originally, it is measured by counting the number of the shortest paths passing through a certain node. A sensor node is identified as a critical node if upon its removal, more sensor nodes will have no path of length $\leq l_{\max}$ to a sink, where $l_{\max}$ is the maximum acceptable path length. Using l-CRC, a sensor node is critical if its centrality index is above a threshold. l-CRC is formulated as

$$l\text{-CRC}(v) = <l\text{-CC}(v), l\text{-RC}(v)> \quad (1)$$

l-CRC has two values: length-constrained connectivity centrality (l-CC) and length-constrained rerouting centrality (l-RC), which are formulated below.

$$l\text{-CC}(v) = |\{w \in D(v);\ d(w,S) \leq l_{\max}, d_v(w,S) > l_{\max}\}| \quad (2)$$

$$l\text{-RC}(v) = \sum_{w \in D(v)} \left(\frac{\max\{d_v(w,S), l_{\max}\}}{\max\{d(w,S), l_{\max}\}} - 1 \right);\ d_v(w,S) \neq \infty \quad (3)$$

$D(v)$ is the set of node v's descendants in the routing tree, S is the set of sinks, $d(u,w)$ denotes the shortest path length between nodes u and w, while $d_v(u,w)$ represents the length of the shortest path from u to w which does not visit v. After identifying the critical nodes, relays are deployed around those nodes to preserve backup paths if they die. Using l-CRC allows us to trade off the deployment cost against the robustness of the network. GRASP-ABP has been shown to deploy fewer relays compared to the algorithm in [2, 9].

The greedy randomized adaptive search procedure (GRASP) [4] is a metaheuristic intended to capture the good features of pure greedy algorithms and of random construction procedures. It is an iterative process, which consists of two phases: a construction phase and a local search phase. The construction phase builds a feasible solution as a good starting solution for the local search. The probabilistic component of a GRASP is characterised by randomly choosing one of the best initial solution. Since the solution produced by the construction phase is not necessarily the local optimum, the local search works iteratively to replace the current solution by a better one.

3 MULTIPLE SINK PLACEMENT (MSP)

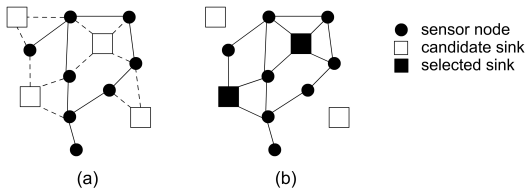

Figure 1. (a) A WSN with 4 candidate sinks and (b) the double-covered WSN where $l_{\max} = 3$

We partition nodes into a set of sensors T and sinks S. In the graph representation, $V = T \cup S$. Note that at this stage we do not use relay nodes yet. A sensor node is *double-covered* if and only if it has at least two paths of length $\leq l_{\max}$ to two sinks in S. If a sensor node is not double-covered, it is *uncovered*. We define a WSN to be double-covered if each sensor node $v \in T$ is double-covered. In the *multiple sink placement* problem, given a graph $G = (T \cup A_{\mathrm{S}}, E)$, where A_{S} is a set of candidate locations for sinks with a non-negative cost function $c : A_{\mathrm{S}} \rightarrow \mathbb{R}$, we find a minimum cost subset $S \subseteq A_{\mathrm{S}}$ such that $H = (T \cup S, E{\downarrow_{T \cup S}})$ is double-covered. We illustrate this problem in Figure 1.

We propose GRASP-MSP to solve the multiple sink placement problem. As with other GRASP-based algorithms, GRASP-MSP consists of two steps: *construction phase* to construct an initial feasible solution and *local search phase* to explore the neighbourhood of the initial solution, looking for lower cost solutions. To speed-up our GRASP algorithm's processing time, we compute the shortest path from all sensor nodes to all candidate sinks once in the beginning and store the length of the shortest path in *Distance* table, while the parent of each sensor node in the shortest path to a candidate sink is stored in *Parent* table. For example, for $G = (T \cup A_{\mathrm{S}}, E)$, $\mathit{Distance}_G(v,w)$ shows the length of the shortest path from a sensor node $v \in T$ to a candidate sink $w \in A_{\mathrm{S}}$, while $\mathit{Parent}_G(v,w)$ shows the parent of a sensor node v on the shortest path to a candidate sink $w \in A_{\mathrm{S}}$.

3.1 GRASP-MSP: Construction Phase

In the construction phase, we find S, an initial set of sinks. Instead of selecting the best candidate sink from A_{S} to be put in S, which can minimise the number of uncovered nodes, we add randomisation to the initial solution by choosing a sink from A_{S} randomly. This random selection is repeated until the network is double-covered or all candidate sinks have been chosen.

3.2 GRASP-MSP: Node-based Local Search

Let S be the set of sinks. We explore the neighbourhood of the current solution by adding a new sink $s \in A_{\mathrm{S}} \backslash S$ into S that can eliminate some existing sinks from S to minimise the total cost as possible. This move must always ensure that the network is double-covered.

3.3 GRASP-MSP: Algorithm Description

The GRASP-MSP pseudocode is given in Algorithm 1. It takes as input the original graph $G=(T\cup A_{\mathrm{S}}, E)$, the set T of sensor nodes, the set A_{S} of candidate sinks, the cost function c, the pre-computed $Distance_G$ table, the maximum acceptable path length $l_{\max}$, and the number of iterations (*max_iterations*). In each iteration, the construction phase to find the initial set of sinks S is executed in line 3. The local search starts with the initialisation of the best set and the best cost in line 6. The loop from line 7 to 22 searches for the best move, i.e. finding a new sink $r\in A_{\mathrm{S}}\setminus S$ that can eliminate as many existing sinks from S as possible. The loop from line 9 to 15 tries to find the set $Z\subseteq S$ of existing sinks that are safe to be removed after the insertion of r. The sinks in Z are safe to be removed if all sensor nodes in $H=(T\cup S\cup\{r\}\setminus Z, E{\downarrow}_{T\cup S\cup\{r\}\setminus Z})$ are double-covered. In line 16, we check if the new solution reduces the total cost of the current best solution. If the total cost can be reduced, we reset the set of the best set in line 17. If the total cost is the same, we keep this new solution in the set of the best set as shown from line 19 to 21.

When all moves have been evaluated, we check in line 23 if an improving solution has been found. If the moves produce a better solution, the set of sinks S is updated in line 24 by selecting one best set randomly from the set of the best set. Then, the local search continues. If at the end of the local search we find a better solution compared to the best solution found so far, we update in line 29 the set of sinks and the lowest total cost found. The best sink set S^* is returned in line 32.

Algorithm 1. GRASP-MSP

Input : $G, T, A_{\mathrm{S}}, c, Distance_G, l_{\max}, max\_iterations$
Output: S^*

1: $best\_value\leftarrow\infty$
2: **for** $i\leftarrow 1$ to *max_iterations* **do**
/* Construction phase */
3: Find S by choosing sinks from A_{S} randomly
4: **do**
5: $solution\_updated\leftarrow$ **false**
/* Local search phase */
6: $best\_set_0\leftarrow S$, $best\_cost\leftarrow\sum_{v\in S} c_v$, $best\_num\_set\leftarrow 1$
7: **for** all $r\in A_{\mathrm{S}}\setminus S$ **do**
8: $Z\leftarrow\emptyset$
9: **for** all $t\in S$ **do**
10: $Z\leftarrow Z\cup\{t\}$, $H=(T\cup S\cup\{r\}\setminus Z, E{\downarrow}_{T\cup S\cup\{r\}\setminus Z})$
11: Find *num_uncovered* in H using $Distance_G$ and $l_{\max}$
12: **if** $num\_uncovered>0$ **then**
13: $Z\leftarrow Z\setminus\{t\}$
14: **end if**
15: **end for**
16: **if** $\sum_{v\in S\cup\{r\}\setminus Z} c_v < best\_cost$ **then**
17: $best\_num\_set\leftarrow 0$
18: **end if**
19: **if** $\sum_{v\in S\cup\{r\}\setminus Z} c_v \le best\_cost$ **and** $S\cup\{r\}\setminus Z\notin best\_set$ **then**
20: $best\_set_{best\_num\_set}\leftarrow S\cup\{r\}\setminus Z$, $best\_cost\leftarrow\sum_{v\in S\cup\{r\}\setminus Z} c_v$, $best\_num\_set$++
21: **end if**
22: **end for**
23: **if** $best\_cost<\sum_{v\in S} c_v$ **then**
24: $S\leftarrow$ select a set randomly from *best_set*
25: $solution\_updated\leftarrow$ **true**
26: **end if**
27: **while** *solution_updated*
/* Best solution update */
28: **if** $\sum_{v\in S} c_v < best\_value$ **then**
29: $S^*\leftarrow S$, $best\_value\leftarrow\sum_{v\in S} c_v$
30: **end if**
31: **end for**
32: **return** S^*

3.4 Evaluation of GRASP-MSP

In the evaluation, we want to show that GRASP-MSP is effective and efficient in finding the lowest cost sink deployment compared to other closely-related algorithms and the optimal solution. To measure the performance of the algorithms, we use cost and runtime metrics. Cost measures the total cost of sinks, while runtime is the algorithm's total execution time.

The results are based on the mean value of 20 randomly generated network deployments. The network consists of 100 sensor nodes deployed within randomly perturbed grids, where a sensor node is placed in a unit grid of 8m $\times$ 8m and the coordinates are perturbed. To get sparse networks (average degree 2-3), we generate more grid points than the number of nodes. We use 11×11 grids to randomly deploy 100 sensor nodes. 25 candidate sinks are also distributed in a grid area, where a candidate occupies a unit grid of 18m $\times$ 18m. Both sensor nodes and sinks use 10 metres transmission range.

We compare GRASP-MSP against *minimise the number of sinks for fault-tolerance* (MSFT) algorithm, *cluster-based sampling for multiple sink placement* (CBS-MSP), the greedy version of multiple sink placement (Greedy-MSP) and the optimal solution. MSFT and CBS-MSP are algorithms based on the well-known k-means clustering algorithm. MSFT is similar to [8]. In [8], sinks can be deployed anywhere and they are added one by one in the network until a required lifetime is met. Unlike [8], MSFT has candidate locations and keeps adding sinks until the network is double-covered. CBS-MSP is similar to *cluster-based sampling* (CBS) algorithm proposed in [3], but with some modification. In CBS, the number of sinks is given and the objective is to minimise the total road distance from all nodes to the sinks where each node is required to be double-covered. Unlike CBS, CBS-MSP tries to reduce the number of sinks and thus the deployment cost. We implement CBS-MSP using path length to represent distance between two nodes and also we have path-length restrictions. In each iteration, both CBS and CBS-MSP try to find the best sink locations to ensure the network is double-covered. k-means clustering algorithm is used in these algorithms to divide the network into clusters and to find the position of each sink, which is in the centre of a cluster. The performances of MSFT and CBS-MSP depend on the randomly selected sink locations. Therefore, we use the maximum iteration (*MaxIter*) to limit the number of iterations. Greedy-MSP is similar to [11], but it considers double-covered networks. Greedy-MSP deploys sinks one by one until the network is double-covered. In each iteration, the greedy move picks the best sink that can minimise the number of uncovered nodes as possible. In Greedy-MSP, if two or more moves offer the same solution, we select one arbitrarily. We also try to evaluate them all, which we call Greedy-MSP-All.

The optimal solution is modeled using binary linear programming with the objective is to minimise the total sink cost, i.e.

$$\min \sum_{j\in S} c_j x_j \tag{4}$$

subject to the following constraints

$$\sum_{j\in S} l_{ij} x_j \ge 2;\ \forall i\in T \tag{5}$$

$$d_{ij}\le l_{\max}\Rightarrow l_{ij}=1,\ d_{ij}>l_{\max}\Rightarrow l_{ij}=0;\ i\in T,\ j\in S \tag{6}$$

$$x_j\in\{0,1\};\ j\in S \tag{7}$$

c is the cost of a candidate sink x. The first constraint guarantees each sensor node has at least two paths to two sinks. The paths are length-bounded, which are shown in the second constraint using a binary

function l_{ij}. It has value equal to one if the shortest path length from a sensor node i to a sink $j \leq l_{\max}$, otherwise its value is zero. In the third constraint, a candidate sink is either selected to be deployed or not. The binary linear programming for the optimal solution is implemented in Matlab, while the other algorithms are written in C++. Tests are carried out in 2.40 GHz Intel Core2 Duo CPU with 4 GB of RAM.

In the simulation, we find the best locations to deploy the least number of sinks to make the networks double-covered. We consider the cases where the maximum acceptable path length from each sensor node to a sink is 6 and 10. The number of sinks deployed by each algorithm is shown in Figure 2 and the runtime is in Table 1. The simulation results show that GRASP-MSP with *max_iterations* $= 10$ requires the same number of sinks with shorter runtime compared to the optimal solution. It also outperforms MSFT, CBS-MSP and Greedy-MSP. Greedy-MSP has the shortest runtime, but it places more sinks compared to GRASP-MSP. At this stage, finding the optimal solution is sufficient for the multiple sink placement problem since the runtime issue is not a problem in offline algorithms. However, using local search technique to later solve the multiple sink and relay placement problem is fully justified by GRASP-MSP as it provides the same result as the optimal solution and even faster.

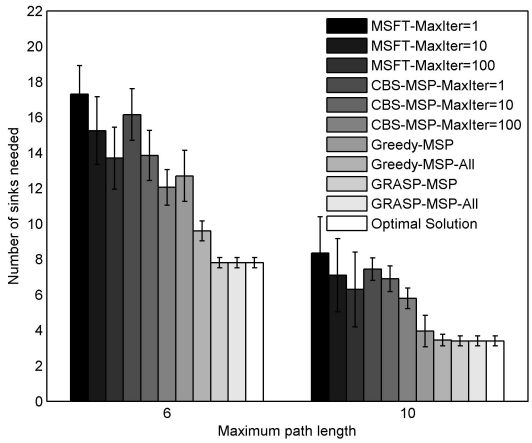

Figure 2. Number of sinks needed for multiple sink placement algorithms versus maximum path length

Table 1. Multiple sink placement algorithms' runtime

Algorithms	Runtime (sec)	
	$l_{\max} = 6$	$l_{\max} = 10$
MSFT-MaxIter=1	0.4188	0.0288
MSFT-MaxIter=10	4.0429	0.3102
MSFT-MaxIter=100	40.8313	3.1647
CBS-MSP-MaxIter=1	1.0297	0.0235
CBS-MSP-MaxIter=10	10.1797	0.2069
CBS-MSP-MaxIter=100	97.0899	2.0844
Greedy-MSP	0.0024	0.0040
Greedy-MSP-All	7.9664	0.0071
GRASP-MSP	0.0688	0.0452
GRASP-MSP-All	0.1305	0.0750
Optimal Solution	0.0727	0.0867

4 MULTIPLE SINK AND RELAY PLACEMENT (MSRP)

For the sink and relay placement, nodes are partitioned into a set of sensors T, relays R and sinks S. In the graph representation, $V = T \cup R \cup S$. We identify a sensor node to be *critical* if and only if upon its removal, more sensor nodes will have no path of length $\leq$

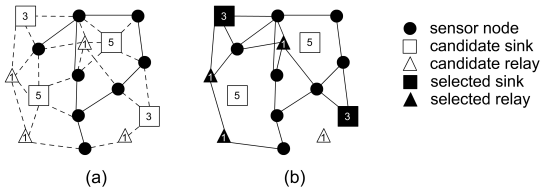

Figure 3. (a) A WSN with 4 candidate sinks and 4 candidate relays, and (b) the double-covered and noncritical WSN where $l_{\max} = 3$

$l_{\max}$ to a sink. Otherwise, it is *noncritical*. We define a WSN to be noncritical if each sensor node $v \in T$ is noncritical. In the *multiple sink and relay placement* problem, given a graph $G = (T \cup A_{\mathrm{R}} \cup A_{\mathrm{S}}, E)$, where A_{R} and A_{S} are sets of candidate locations for relays and sinks, respectively, we find minimum cost subsets $R \subseteq A_{\mathrm{R}}$ and $S \subseteq A_{\mathrm{S}}$ such that $H = (T \cup R \cup S, E \downarrow_{T \cup R \cup S})$ is double-covered and noncritical. The relay and sink candidate locations are associated with a non-negative cost function $c : A_{\mathrm{R}} \cup A_{\mathrm{S}} \rightarrow \mathbb{R}$. We assume that a relay is cheaper than a sink because sinks usually are assumed to be powered and have WiFi/ethernet backhaul. The multiple sink and relay placement problem is illustrated in Figure 3, where the numbers represent the devices' costs.

We develop GRASP-MSRP to solve the multiple sink and relay placement problem. To identify critical nodes, GRASP-MSRP uses length-constrained connectivity and rerouting centrality (l-CRC) proposed in [10]. A node is critical if its centrality index is above a threshold. We can raise the threshold to trade off the deployment cost against the robustness of the network. However, in this paper, we only assume zero threshold for full reliability.

4.1 GRASP-MSRP: Construction Phase

In the construction phase, we find R and S as our initial sets of relays and sinks, respectively. We need at least two sinks for a double-covered WSN, so we firstly choose two sinks randomly from A_{S}. We then deploy relays from A_{R} to minimise the number of uncovered and critical nodes. If a sensor node $v \in T$ is uncovered, we try to place some relays to construct a path to a sink $w \in S$ if *Distance*$_H(v, w) > l_{\max}$ but *Distance*$_G(v, w) \leq l_{\max}$. We choose the relays that appear on the shortest path from v to w by tracing the path in *Parent*$_G(v, w)$. If the sensor node needs two paths to make it double-covered, this step is repeated twice. If a sensor node $v \in T$ is critical, we deploy relays that appear on the shortest path from each descendant of v to a sink $w \in S$ bypassing v, as long as the shortest path length is $\leq l_{\max}$. We basically alternate the sink and relay addition during this process. However, we do not add more sinks if at some points the network is already double-covered. If the problem has a feasible solution, the network is double-covered and noncritical at the end of the construction phase.

4.2 GRASP-MSRP: Node-based Local Search

Let R be the set of relays and S be the set of sinks. We look for a lower cost solution by adding either a new relay $r \in A_{\mathrm{R}} \setminus R$ into R or a new sink $s \in A_{\mathrm{S}} \setminus S$ into S that can eliminate some existing relays from R and sinks from S to minimise the total cost as possible. Given that the cost of a sink is higher than the cost of a relay, we also try to minimise the total cost by adding some relays into R when we eliminate an existing sink from S. The local search moves are performed to reduce the total cost, but must ensure that the network is still double-covered and noncritical in each iteration.

4.3 GRASP-MSRP: Algorithm Description

The GRASP-MSRP pseudocode is given in Algorithm 2. Its concept is similar to GRASP-MSP in Algorithm 1 with some differences. We will only describe its differences. Firstly, GRASP-MSRP takes the set of candidate relays A_{R} as one of its input. Secondly, the shortest paths from all sensor nodes to all sinks are computed several times in line 12, 18 and 24 due to the addition and elimination of relays.

Algorithm 2. GRASP-MSRP

```
Input  : G, T, A_R, A_S, c, l_max, max_iterations
Output: R*, S*
 1:  best_value ← ∞
 2:  for i ← 1 to max_iterations do
                                          /* Construction phase */
 3:      Find initial R and S,  W ← R∪S
 4:      do
 5:          solution_updated ← false
                                          /* Local search phase */
 6:          best_set_0 ← W,  best_cost ← Σ_{v∈W} c_v,  best_num_set ← 1
 7:          for all r ∈ A_R ∪ A_S \ W do
 8:              Y ← {r},  Z ← ∅,  X ← ∅
 9:              for all t ∈ W ∪ X do
10:                  Z ← Z ∪ {t},  X ← ∅
11:                  H = (T ∪ W ∪ Y \ Z, E↓_{T∪W∪Y\Z})
12:                  Calculate Distance_H
13:                  Find uncovered_set in H using Distance_H and l_max
14:                  if |uncovered_set| > 0 then
15:                      X ← X ∪ {Find relays to minimise |uncovered_set|}
16:                  end if
17:                  H = (T ∪ W ∪ X ∪ Y \ Z, E↓_{T∪W∪X∪Y\Z})
18:                  Calculate Distance_H
19:                  Find critical_set in H using Distance_H and l_max
20:                  if |critical_set| > 0 then
21:                      X ← X ∪ {Find relays to minimise |critical_set|}
22:                  end if
23:                  H = (T ∪ W ∪ X ∪ Y \ Z, E↓_{T∪W∪X∪Y\Z})
24:                  Calculate Distance_H
25:                  Calculate num_uncovered and num_critical in H
                     using Distance_H and l_max
26:                  if num_uncovered = 0 and num_critical = 0 then
27:                      Y ← Y ∪ X,  Z ← Z \ X
28:                  end if
29:                  if num_uncovered > 0 or num_critical > 0 then
30:                      Z ← Z \ {t}
31:                  end if
32:              end for
33:              if Σ_{v∈W∪Y\Z} c_v < best_cost then
34:                  best_num_set ← 0
35:              end if
36:              if Σ_{v∈W∪Y\Z} c_v ≤ best_cost and W ∪ Y \ Z ∉ best_set then
37:                  best_set_{best_num_set} ← W ∪ Y \ Z,
                     best_cost ← Σ_{v∈W∪Y\Z} c_v,
                     best_num_set++
38:              end if
39:          end for
40:          if best_cost < Σ_{v∈W} c_v then
41:              W ← select a set randomly from best_set
42:              solution_updated ← true
43:          end if
44:      while solution_updated
                                          /* Best solution update */
45:      if Σ_{v∈W} c_v < best_value then
46:          R* ← ∅,  S* ← ∅
47:          for all v ∈ W do
48:              if v ∈ A_R then
49:                  R* ← R* ∪ {v}
50:              else
51:                  S* ← S* ∪ {v}
52:              end if
53:          end for
54:          best_value ← Σ_{v∈W} c_v
55:      end if
56:  end for
57:  return R*, S*
```

In line 13, the algorithm checks for uncovered nodes. If some exist, it tries to deploy relays in line 15. The identification of critical nodes is performed in line 19. If some exist, relays are added in line 21. Note that we try to minimise the total cost by adding some relays when we eliminate a sink. These relays are saved in X as shown in line 15 and 21, which later will be included in Y, the set of new relays and sinks to be inserted, if X helps the network become double-covered and noncritical. The network is checked if it is double-covered and noncritical in line 25. The rest of the pseudocode has similar role to GRASP-MSP's.

4.4 Evaluation of GRASP-MSRP

We evaluate the total deployment cost and the runtime of GRASP-MSRP against *minimise the number of sinks and relays for fault-tolerance* (MSRFT) algorithm, *cluster-based sampling for multiple sink and relay placement* (CBS-MSRP) and the greedy version of multiple sink and relay placement (Greedy-MSRP). MSRFT, CBS-MSRP and Greedy-MSRP use MSFT, CBS-MSP and Greedy-MSP, respectively, to find the best locations to deploy sinks and GRASP-ABP [10] to deploy relays. These three algorithms start by finding the best locations for two sinks before utilising GRASP-ABP [10] to deploy relays. The number of sinks is gradually increased and GRASP-ABP is used to deploy relays until the network becomes double-covered and noncritical.

We follow the same simulation setting as for the multiple sink placement problem in Section 3.4. In addition, we have 81 candidate relays distributed evenly in a grid area. A candidate relay occupies a unit grid of 10m $\times$ 10m. In the simulation, we only use 100 as the maximum iteration (*MaxIter*) for MSRFT and CBS-MSRP. We also use different sink costs (c_{S}), i.e. 3, 6, randomly between 3 and 6, and 10 units, while relay cost is 1 unit. The total sink and relay deployment cost suggested by each algorithm with $l_{\max} = 6$ is shown in Figure 4 and the runtime is in Table 2. The results show that GRASP-MSRP with *max_iterations*$= 10$ has the lowest total cost compared to other algorithms. Although GRASP-MSRP's runtime is the longest, this is acceptable since the deployment planning is an offline process, which is carried out during the initial design phase.

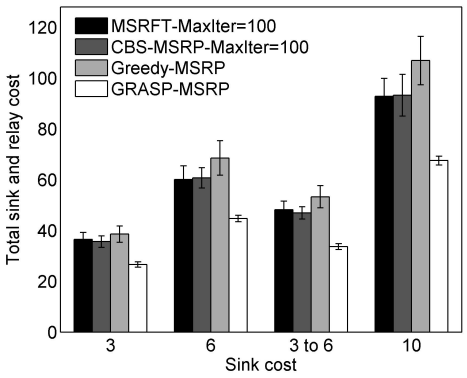

Figure 4. Total cost for multiple sink and relay placement algorithms versus sink cost

Table 2. Multiple sink and relay placement algorithms' runtime

Algorithms	Runtime (sec)			
	$c_{\mathrm{S}} = 3$	$c_{\mathrm{S}} = 6$	$c_{\mathrm{S}} = 3-6$	$c_{\mathrm{S}} = 10$
MSRFT-MaxIter=100	61.9235	60.3157	60.1228	59.2399
CBS-MSRP-MaxIter=100	61.2929	64.0727	62.0789	61.9352
Greedy-MSRP	153.7541	146.8796	130.1220	141.8679
GRASP-MSRP	196.5039	216.4508	251.2635	228.3422

The numbers of deployed sinks and relays for GRASP-MSRP are depicted in Figure 5. The results show that more sinks are exchanged with relays when the sink cost increases to reduce the total deployment cost. We also simulate GRASP-MSRP with $l_{\max} = 10$ as shown in Figure 6. When we increase $l_{\max}$, the number of sinks decreases significantly but the number of relays does not increase much. The simulation runtime for $l_{\max} = 6$ is 196.5039 seconds and for $l_{\max} = 10$ is 348.6041 seconds.

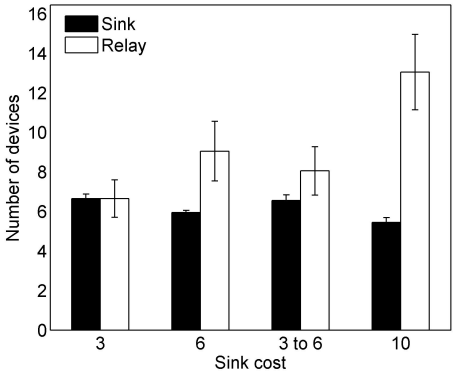

Figure 5. Total numbers of sinks and relays for GRASP-MSRP versus sink cost

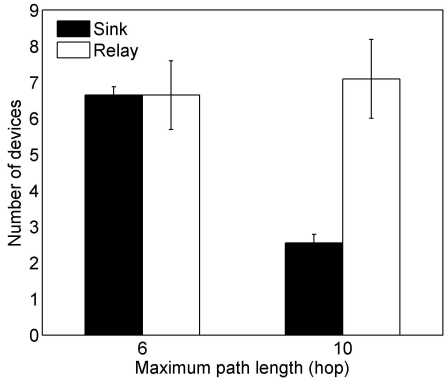

Figure 6. Total numbers of sinks and relays for GRASP-MSRP versus maximum path length

5 CONCLUSION

We have studied the WSN deployment planning problem where the aim is to protect the network against one single failure, of either a sink or a sensor node. We design a network to be double-covered and noncritical. Double-covered means each sensor node must have at least two paths with acceptable length to two sinks. Noncritical means all sensor nodes must have length-bound paths to sinks when an arbitrary sensor node fails. We propose GRASP-MSP and GRASP-MSRP, local search algorithms based on the GRASP technique to minimise the total cost of deployment. GRASP-MSP solves the multiple sink placement problem by ensuring that each sensor node in the network is double-covered. We demonstrate empirically that it achieves the same deployment cost as the optimal solution with shorter runtime. GRASP-MSP's simulation results justify the use of local search to solve the multiple sink and relay placement problem, where linear solution is not available. GRASP-MSRP tries to optimise the multiple sink and relay placement problem. Compared to the most closely-related algorithms, GRASP-MSRP sacrifices runtime to achieve the lowest total cost. This is not a significant problem in the network deployment planning because it is an offline process. We are currently investigating the performance of the topologies generated by our proposed algorithms using a network simulator.

ACKNOWLEDGEMENTS

This research is fully funded by the NEMBES project, supported by the Irish Higher Education Authority PRTLI-IV research program.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, 'Wireless Sensor Networks: A Survey', *Computer Networks*, **38**(4), 393–422, (Mar. 2002).

[2] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus, 'Deploying Sensor Networks with Guaranteed Capacity and Fault Tolerance', in *Proc. 6th ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc'05)*, pp. 309–319, (May 2005).

[3] H. Cambazard, D. Mehta, B. O'Sullivan, L. Quesada, M. Ruffini, D. Payne, and L. Doyle, 'A Combinatorial Optimisation Approach to the Design of Dual-Parented Long-Reach Passive Optical Networks', in *Proc. 23rd IEEE Int'l Conf. Tools with Artificial Intelligence (ICTAI'11)*, pp. 785–792, (Nov. 2011).

[4] T. A. Feo and M. G. C. Resende, 'Greedy Randomized Adaptive Search Procedures', *Journal of Global Optimization*, **6**, 109–133, (1995).

[5] X. Han, X. Cao, E. L. Lloyd, and C. C. Shen, 'Fault-tolerant Relay Node Placement in Heterogeneous Wireless Sensor Networks', *IEEE Transactions on Mobile Computing*, **9**(5), 643–656, (May 2010).

[6] S. Mahmud, H. Wu, and J. Xue, 'Efficient Energy Balancing Aware Multiple Base Station Deployment for WSNs', in *Proc. 8th European Conf. Wireless Sensor Networks (EWSN'11)*, pp. 179–194, (Feb. 2011).

[7] S. Misra, S. D. Hong, G. Xue, and J. Tang, 'Constrained Relay Node Placement in Wireless Sensor Networks to Meet Connectivity and Survivability Requirements', in *Proc. 27th Ann. IEEE Conf. Computer Communications (INFOCOM'08)*, pp. 281–285, (Apr. 2008).

[8] E. I. Oyman and C. Ersoy, 'Multiple Sink Network Design Problem in Large Scale Wireless Sensor Networks', in *Proc. IEEE Int'l Conf. Communications (ICC'04)*, pp. 3663–3667, (Jun. 2004).

[9] J. Pu, Z. Xiong, and X. Lu, 'Fault-Tolerant Deployment with k-connectivity and Partial k-connectivity in Sensor Networks', *Wireless Communications and Mobile Computing*, **9**(7), 909–919, (May 2008).

[10] L. Sitanayah, K. N. Brown, and C. J. Sreenan, 'Fault-Tolerant Relay Deployment Based on Length-Constrained Connectivity and Rerouting Centrality in Wireless Sensor Networks', in *Proc. 9th European Conference on Wireless Sensor Networks (EWSN'12)*, pp. 115–130, (Feb. 2012).

[11] X. Xu and W. Liang, 'Placing Optimal Number of Sinks in Sensor Networks for Network Lifetime Maximization', in *Proc. IEEE Int'l Conf. Communications (ICC'11)*, (Jun. 2011).

[12] W. Youssef and M. Younis, 'Intelligent Gateway Placement for Reduced Data Latency in Wireless Sensor Networks', in *Proc. IEEE Int'l Conf. Communications (ICC'07)*, pp. 3805–3810, (Jun. 2007).